c J C

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

ATTNY. DOCKET: 085804-010200
PATENT NO.: 7,051,119
APPLICANT: Thomas J. Shafron et al.
SERIAL NO.: 09/904,300
FILING DATE: July 12, 2001
TITLE: **METHOD AND SYSTEM FOR ENABLING A SCRIPT ON A FIRST COMPUTER TO COMMUNICATE AND EXCHANGE DATA WITH A SCRIPT ON A SECOND COMPUTER OVER A NETWORK**

REGULAR MAIL CERTIFICATE

Date of Deposit: October 26, 2006

I hereby certify that the following attached paper(s) and/or fee

(1) Request for Certificate of Correction Under 37 C.F.R. §1.322 (3 pgs);
(2) Certificate of Correction form (PTO/SB/44) (1 pg);
(3) Copy of the Information Disclosure Statement, signed by the Examiner (2 pgs);
(4) Copy of a postcard indicating receipt of the Information Disclosure Statement and the two cited references at the U.S. Patent and Trademark Office on October 7, 2005 (1 pg);
(5) Copy of facsimile as sent by patentees' attorneys along with a copy of a "Message Confirmation" sheet indicating receipt thereof by the Examiner on October 18, 2005 (2 pgs);
(6) Copies of Two References as cited:
- VIZIR: An Integrated Environment for Distributed Program Visualization, Hao et al., Durham, North Carolina, Jan. 1995, p. 288-292
- Concurrent Application Control in Collaborative Computing, Hao et al., HPL-94-37, Hewlett-Packard Company, 1994, p. 1-12 ;
(7) Copy of the Notice of Allowability, including the Examiner's remarks (3 pgs); and
(8) A self-addressed stamped postcard, return of which is requested to acknowledge receipt of the enclosed documents.

are being deposited with the United States Postal Service Regular Mail Post Office to Addressee service under 37 C.F.R. Section 1.10 on the date indicated above and is addressed to Attn: Certificate of Correction Branch, Commissioner for Patents, P.O. Box 1450; Alexandria, VA 22313-1450.

Respectfully submitted,
GREENBERG TRAURIG, LLP.
Girlene Banks
Girlene Banks
Legal Assistant

Dated: October 26, 2006

CORRESPONDENCE:

James J. DeCarlo, Esq.
GREENBERG TRAURIG, LLP
MetLife Building
200 Park Avenue
New York, NY 10166
Tele: (212) 801-6729
Fax: (212) 801-6400

NY 238205510v1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent of:	Shafron et al.		
Patent No.	7,051,119	Application No.:	09/904,300
Issue Date:	May 23, 2006	Filing Date:	July 12, 2001
Title:	METHOD AND SYSTEM FOR ENABLING A SCRIPT ON A FIRST COMPUTER TO COMMUNICATE AND EXCHANGE DATA WITH A SCRIPT ON A SECOND COMPUTER OVER A NETWORK		

Attn: Certificate of Correction Branch
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

REQUEST FOR CERTIFICATE OF CORRECTION UNDER 37 C.F.R. § 1.322

Sir:

The applicants' undersigned attorney has reviewed the above-identified patent and found errors therein. A Certificate of Correction form (PTO/SB/44) is submitted herewith stating as follows:

The following references should appear on the front page of the patent under the heading OTHER PUBLICATIONS:

--Concurrent Application Control in Collaborative Computing, Hao et al., HPL-94-37, Hewlett-Packard Company, 1994, p. 1-12.

VIZIR: An Integrated Environment for Distributed Program Visualization, Hao et al., Durham, North Carolina, Jan. 1995, p. 288-292.--

The two references noted above were listed in the "Non-Patent Literature Documents" section of an Information Disclosure Statement (PTO/SB08B) filed in the above-identified

application on October 7, 2005. A copy of the Information Disclosure Statement, signed by the Examiner, is enclosed.

Also enclosed is a copy of a postcard indicating receipt of the Information Disclosure Statement and the two cited references at the U.S. Patent and Trademark Office on October 7, 2005.

The Information Disclosure Statement form (PTO/SB08B) was signed by the Examiner on October 18, 2005, with no indication whether the two references were considered by the Examiner. However, the Examiner evidently requested copies of the two references on October 18, 2005, as shown by the facsimile transmission cover sheet sent from patentees' attorneys to the Examiner's direct facsimile number (571-273-4004). This facsimile transmission included copies of the above-noted references, with the references and the cover sheet totaling 20 pages. A copy of this facsimile as sent by patentees' attorneys (20 pages) is also enclosed, along with a copy of a "Message Confirmation" sheet indicating receipt thereof by the Examiner on October 18, 2005.

Of further note, in the Notice of Allowability dated October 18, 2005, Examiner Jacobs wrote at page 2:

> <u>Information Disclosure Statement</u>
>
> 1. The information disclosure statement (IDS) submitted on October 7, 2005 was filed after the mailing date of the Notice of Allowance on August 26, 2005. The submission is in compliance with the provisions of 37 C.F.R. 1.97. Accordingly, the information disclosure statement is being considered by the examiner.

A copy of the Notice of Allowability, including the Examiner's remarks, is also enclosed.

Based on the documents described just above, the two cited references should be made of record in the present patent. Issuance of a Certificate of Correction, adding these references to the "References Cited" in the patent, is therefore respectfully requested.

No fees are believed to be due. The Commissioner nevertheless is hereby authorized to charge any applicable fees to Deposit Account No. 50-1561 of Greenberg Traurig, LLP.

The applicant's attorney may be reached by telephone at 212-801-6729. All correspondence should continue to be directed to the address given below, which is the address associated with Customer Number 32361.

Respectfully submitted,

Date: October 26, 2006

James DeCarlo
Attorney for Patentees
Registration Number 36,120

GREENBERG TRAURIG, LLP
200 Park Avenue
New York, NY 10166

NY 238202906v2 10/26/2006

NOV 03 2006

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

Page 1 of 1

PATENT NO. : 7,051,119

APPLICATION NO.: 09/904,300

ISSUE DATE : May 23, 2006

INVENTOR(S) : Shafron et al.

It is certified that an error appears or errors appear in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

The following references should appear on the front page of the patent under the heading OTHER PUBLICATIONS:

--Concurrent Application Control in Collaborative Computing, Hao et al., HPL-94-37, Hewlett-Packard Company, 1994, p. 1-12.

VIZIR: An Integrated Environment for Distributed Program Visualization, Hao et al., Durham, North Carolina, Jan. 1995, p. 288-292.--

MAILING ADDRESS OF SENDER (Please do not use customer number below):

GREENBERG TRAURIG, LLP
200 Park Avenue
New York, NY 10166

NOV 03 2008

OIPE IAP8
OCT 0 7 2005
PATENT & TRADEMARK OFFICE

PTO/SB/08A (07-05)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	09/904,300
Filing Date	July 12, 2001
First Named Inventor	Thomas J. Shafron
Art Unit	2157
Examiner Name	LaShonda T. Jacobs
Attorney Docket Number	85804.010200

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number: Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
LJ		US- 6,128,649	10-03-2000	Smith et al.	
LJ		US- 6,125,385	09-26-2000	Wies et al.	
LJ		US- 5,996,003	11-30-1999	Namikata et al.	
LJ		US- 5,708,780	01-13-1998	Levergood et al.	
LJ		US- 5,446,842	08-29-1995	Schaeffer et al.	
LJ		US- 6,826,595	11-30-2004	Barbash et al.	
LJ		US- 6,356,283	03-12-2002	Guedalia, Joshua Siegfried	
LJ		US- 5,844,553	12-01-1998	Hao et al.	
LJ		US- 5,790,818	08-04-1998	Martin, Rocco	
LJ		US- 2004/0165007	08-26-2004	Shafron, Thomas J.	
LJ		US- 6,268,852	07-31-2001	Lindhorst et al.	
LJ		US- 6,112,242	08-29-2000	Jois et al.	
LJ		US- 5,999,912	12-07-1999	Wodarz et al.	
LJ		US- 5,355,472	10-11-1994	Lewis, Jonathan R.T.	
		US-			
		US-			
		US-			
		US-			
		US-			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document: Country Code[3] Number[4] Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]
LJ		WO 03/012668 A1	02-13-2003	Yahoo! Inc.	(English Original)	

Examiner Signature	LaShonda Jacobs	Date Considered	10/18/05

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1]Applicant's unique citation designation number (optional). [2]See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3]Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4]For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5]Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6]Applicant is to place a check mark here if English language Translation is attached.

PTO/SB/08B (07-05)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete If Known	
Application Number	09/904,300
Filing Date	July 12, 2001
First Named Inventor	Thomas J. Shafron
Art Unit	2157
Examiner Name	LaShonda T. Jacobs
Attorney Docket Number	85804.010200

Sheet 1 of 1

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
LJ		International Preliminary Examination Report for International Application No. PCT/US02/22209 dated 18 May 2003	
		Concurrent Application Control in Collaborative Computing, Hao et al., HPL-94-37, Hewlett-Packard Company, 1994, p1-12	
		VIZIR: An Integrated Environment for Distributed Program Visualization, Hao et al., Durham, North Carolina, Jan. 1995, p 288-292	

Examiner Signature	LaShonda Jacobs	Date Considered	10/18/05

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Applicant: Thomas J. Shafron et al.
Serial No.: 09/904,300
Docket No.: 85804-010200 (694231/0006)
Attorney: James J. DeCarlo

OIPE
OCT 07 2005
PATENT & TRADEMARK OFFICE

Title: METHOD AND SYSTEM FOR ENABLING A SCRIPT ON A FIRST COMPUTER TO COMMUNICATE AND EXCHANGE DATA WITH A SCRIPT ON A SECOND COMPUTER OVER A NETWORK

Receipt is hereby acknowledged by return of this postal card properly stamped by the US PTO for the above noted matter of the following:

(1) Request for Continued Examination (RCE) Transmittal (1 page, in duplicate);
(2) Communicating Transmitting Formal Drawings (1 page);
(3) Formal Drawings (18 sheets);
(4) Supplemental Information Disclosure Statement (2 pages);
(5) PTO/SB/08A (1 page) and PTO/SB/08B (1 page);
(6) Two (2) references cited;
(7) Copy of International Preliminary Examination Report for PCT/US02/22209 (3 pages); and
(8) A self-addressed stamped postcard, return of which is requested to acknowledge receipt of the enclosed documents.

Mailed: October 7, 2005 via Express Mail EV 559699541 US

MESSAGE CONFIRMATION

10/18/2005 10:17
ID=GREENBERG TRAURIG

DATE	S,R-TIME	DISTANT STATION ID	MODE	PAGES	RESULT	
10/18	06'50"	USPTO	TX	20	OK	0000

Greenberg Traurig

Transmittal Cover Sheet

From:	Tel:	E-Mail:
Girlene Banks	212-801-6813	banksg@gtlaw.com

To:	Fax No:	Company:	Phone No.:
Examiner Jacobs	571-273-4004	USPTO	571-272-4004

File No.: 85804-010200

Re: Articles

Date: October 18, 2005 10:02 AM

No. Pages: Including Cover Sheet

If you do not receive all pages properly, please call The sender.

Notes:

Per your telephone conversation, attached are the articles listed on the PTO/SB/08B. Please call me to at 212-801-6813.

Thanks.

Greenberg Traurig

Transmittal Cover Sheet

From:	Tel:	E-Mail:
Girlene Banks	212-801-6813	banksg@gtlaw.com

To:	Fax No:	Company:	Phone No.:
Examiner Jacobs	571-273-4004	USPTO	571-272-4004

File No.: 85804-010200

Re: Articles

Date: October 18, 2005 10:02 AM

No. Pages: Including Cover Sheet

If you do not receive all pages properly, please call The sender.

Notes:

Per your telephone conversation, attached are the articles listed on the PTO/SB/08B. Please call me to at 212-801-6813.

Thanks.

Also sent via: ☐ US Mail ☐ Overnight ☐ Messenger ☐ Email ☒ No Other

IEEE Xplore RELEASE 2.1

You requested this document:

» Key

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

1. **VIZIR: an integrated environment for distributed program visualization**
Hao, M.C.; Karp, A.H.; Waheed, A.; Jazayeri, M.;
Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 1995. MASCOTS '95., Pr International Workshop on
18-20 Jan. 1995 Page(s):288 - 292
Abstract:

VIZIR provides an integrated mechanism for on-line debugging, performance analysis, and data visualization parallel applications. The current VIZIR includes: (1) a mechanism to multicast window-based commands, me program execution events among tasks; (2) the ability to select groups of tasks to interact these events; (3) a these events cause errors in the tasks; and (4) ad hoc visualization of distributed arrays using existing visuali

Abstract | Full Text: PDF(456 KB) IEEE CNF

VIZIR: An Integrated Environment for Distributed Program Visualization

Ming C. Hao, Alan H. Karp, Abdual Waheed, Mehdi Jazayeri
{mhao, karp}@hpl.hp.com, waheed@egr.msu.edu
Hewlett-Packard Research Labs, Palo Alto, CA

Abstract

VIZIR provides an integrated mechanism for on-line debugging, performance analysis, and data visualization of message-passing parallel applications. The current VIZIR includes: (1) a mechanism to multicast window-based commands, message passing and program execution events among tasks; (2) the ability to select groups of tasks to interact these events; (3) a means to report when these events cause errors in the tasks; and (4) ad hoc visualization of distributed arrays using existing visualizers.

1.0 Introduction

Distributed computing is emerging as a realistic and cost-effective approach to parallel processing. This trend is becoming popular, largely due to the advances in networking of heterogenous computing resources that allow the users to use the network as a parallel machine. Several users of conventional parallel supercomputers now consider such distributed parallel architectures as an alternative for their applications. However, despite the immense potential of network-based distributed computing, the existing program development tools for such systems are still primitive in terms of their efficacy. Moreover, the problems involved in developing message-passing programs for any parallel architecture are non-trivial. Visualizing the program behavior has proved to be a useful technique for debugging and analyzing the concurrent behavior of parallel programs but the asynchronous nature of distributed systems make the task even more challenging. VIZIR is an integrated visualization and debugging environment that has been developed at Hewlett-Packard Laboratories to meet with these challenges of tool development and to facilitate the task of distributed programming. Its main objective is to use the strengths of network-based computing for sequential/distributed programs and to apply the existing technologies and tools to solve the problems of distributed program development. Due to the nature of the specific issues involved in developing distributed programs as opposed to the development of sequential programs, the manner in which these existing tools and technologies are applied is the key to solve several tool development problems. The design of VIZIR addresses these issues to enable the users to use their favorite debugging tools for debugging their distributed programs. Similarly, it enables the users to add multiple visualization tools of their choice to VIZIR environment to understand the dynamic execution behavior of their applications.

Tool development for programming parallel and distributed systems has been an active area of research but has largely fallen short of the user expectations [10]. In many cases users are not satisfied with the way a particular tool works and the learning curve associated with using it to accomplish a specific task. VIZIR allows the user to use his/her favorite tool by making it a part of the programming environment. In order to realize this environment, following issues are involved:

1. Enabling the programmer's favorite, existing debugging and visualization tools to become a part of this program development environment. This has become an important issue in view of increasing indifference of the users toward "novel" parallel program development tools.
2. Synchronizing and controlling the activities of the debuggers and visualizer to provide a consistent view of program execution to the user.
3. Integrating heterogenous types of visualization tools, such as general-purpose and conventional performance visualization tools across different platforms. This is necessary to address a wide range of requirements of various types of program behavior visualizations, such as application performance, system performance, and program data visualizations.

We have addressed these issues in the design of VIZIR to assist the users to develop distributed programs using the PVM [4] message-passing library for a cluster of workstations. We have enabled several commonly used debugging tools, such as Softbench's Softdebug, Hewlett Packard's DDE, and IBM's XDE. Similarly, we have integrated pop-

0-8186-6902-0/95 $4.00 © 1995 IEEE

ularly used parallel program visualization tools such as ParaGraph in our system, in addition to commercially available visualization tools, such as Matlab and Gnuplot for customized program performance and data visualizations. This paper depicts the architecture and the main features of the VIZIR.

2.0 Architecture of VIZIR

In a distributed programming environment, an application consists of multiple processes running on one or more physical nodes that are distributed in a network. VIZIR executes each of these application processes under the control of an available (and perhaps user's favorite) debugger. One debugger executing a single process presents the same scenario as debugging a single sequential application. The only difference is the message-passing among these otherwise independent processes. Visualization has been recognized as an appropriate technique to represent message-passing and program execution behavior [8]. Several tools have been developed and used to represent various aspects of concurrent program and system behavior [9]. VIZIR enables the use of these visualization tool on-line by providing three major functionalities:

1. multicasting message-passing events among the application processes;
2. integrating visualization tools to represent multiple perspectives of application behavior; and
3. controlling and synchronizing the execution of application processes and visualization tools.

Figure 1 depicts the overall architecture of VIZIR and its functionality. Despite the distributed processes, the environment allows the user to control the configuration and actions of all the distributed application processes and tools. It is important to note that VIZIR is running locally, whereas the other application processes, debuggers, and visualization tools might be running locally or remotely. Therefore, VIZIR acts as a controller for the whole environment which is the key to resolve the problems involved in visualizing distributed application programs. We present the major functions of VIZIR related with distributed program visualization in the following.

2.1 Multicasting Process Events

Parallel program visualization tools have to rely on some mechanism for multicasting events among the concurrent processes, in order to represent this activity graphically. Program code is instrumented and linked with the available communication library to multicast these communication events. However, most of the instrumentation system for parallel programs in general (such as PICL [3]) and distributed programs in particular (such as Xab [1]) perturb the application behavior mainly due to additional message-passing required for generating and communicating the trace data. In the case of Xab, distributed programs using PVM message-passing library need to use the same message-passing calls to accumulate the trace data. VIZIR does not need such explicit message-passing which might double the actual communication cost of an application. Instead, it relies on a Multiple Event Protocol (MEP [7]) to multicast message-passing and program execution events among the application processes. MEP uses inter-client communication primitives that incur less overhead than using PVM calls to receive these events. Additionally, there is no explicit binding between application and VIZIR's multiple event processing and synchronization activities. Whenever an application process executes a particular message-passing function, it sends that event to the event queue of underlying system which can be triggered by VIZIR. Once VIZIR receives the event, it assigns the event a time-stamp and generates a corresponding trace record. This trace record can be further processed and passed on to the visualization tools to dynamically visualize the application behavior.

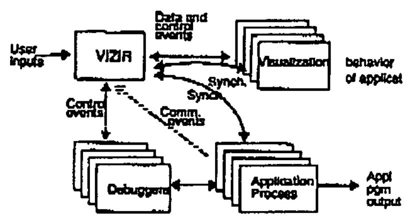


FIGURE 1. Architecture of VIZIR distributed program development and visualization

2.2 Integrating Multiple Tools

Tool integration is a well-known problem in software engineering and there is an on-going standardization effort to develop more practical frameworks for this purpose [2]. More recently, it has found its way into the design of tools for parallel programming because it is difficult for a single tool to satisfy all the requirements of all the users [12]. In order for the tool integration to be useful in case of parallel program development, it should meet two requirements: There should (ideally) be no dependence between the internal semantics of a tool and the rest of the environ-

ment, to ensure generality of the design and to avoid any problems related with the issues of overall performance and portability.

4. It should be possible for the environment to pass necessary performance data and the desired actions to be taken on that data by the tool.

A tool interface (TI) was designed to specifically meet the above two requirements. The functionality of the TI is depicted by Figure 2. As shown in the figure, each visualization tool which is to be integrated into the rest of the environment needs an interface. This interface is used for two specific purposes: (1) receiving data and control information from VIZIR; and (2) forwarding this data and control information to the particular tool that the interface is responsible for. The interface converts the control information in a form which is in accordance with the semantics of that particular tool. Optional bi-directional communication is supported by the interface for synchronization purposes.

VIZIR simplifies the issues involved in tool integration. It provides a common interface to obtain user input to control visualization tools as well as the rest of the environment. VIZIR sends the trace records to the visualizers as soon as they are generated to provide on-line visualization of dynamic program behavior. It can also store the trace data on the local disk as a trace file for the purpose of program replay [5].

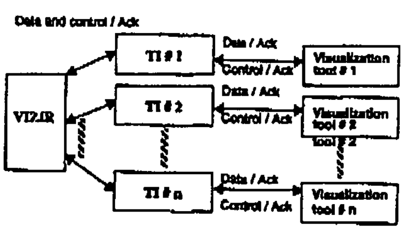


FIGURE 2. Integrating visualization tools into VIZIR for interactive visualization and animation.

2.3 Controlling and Synchronizing

Controlling and synchronizing are perhaps the most important functions of VIZIR that allow the integration of various visualization and debugging tools to work as one programming environment for the user. User can start or stop the execution of the entire distributed application using the control mechanism provided by VIZIR through its GUI. Application processes are executed under the control of debuggers and VIZIR sends the control commands to the multiple debuggers using Event Sense Protocol (ESP [6]). ESP provides a mechanism to multicast window based commands from single control window to some subset of debuggers and visualizers on various processes. It allows VIZIR to control the debuggers without any binding between the two. Therefore, VIZIR environment is efficient, flexible and extensible. Application processes and visualization tools can optionally be synchronized with the VIZIR to ensure the consistency between application behavior and visualization displays. Synchronization has been kept as an option because it is bound to slow down the execution of application processes.

3.0 Features of VIZIR

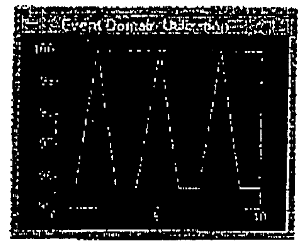
The design features of VIZIR presented in Section 2 have been used to provide several distributed program visualization features. Efficacy of the presently available visualization tools for distributed programs is limited due to the unavailability of these features. This section briefly presents some of these features.

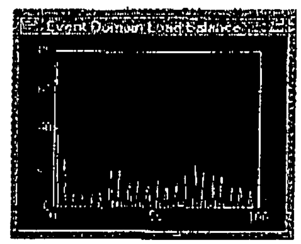
3.1 Consistent Time Stamps

Generation of consistent time stamp has been a non-trivial constraint for existing distributed program visualization tools [1]. VIZIR overcomes this problem due to its architecture that relies on MEP and makes it a global controller for the whole environment. When the applications are running on different workstations, it is not possible to have a notion of consistent global time. VIZIR does not have the application processes to generate the time-stamps with the events. Instead, it only collects and orders the information about event occurrences through MEP and tags the time-stamps. VIZIR orders all incoming events before generating time-stamps, ensuring that they are consistent. Also, there is negligible delay between a communication call and triggering that event using MEP, therefore, the time-stamps are reliable for all practical purposes.

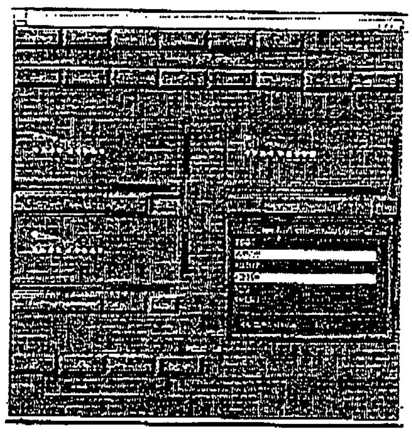
3.2 On-the-Fly Visualization

VIZIR can send the trace records to the visualization tools, immediately after they are generated by assigning time-stamps. Mostly, visualization tools such as ParaGraph process one trace record at a time. As soon as a new trace record is received, all the selected displays are updated by the tool. This process is facilitated by the tool interfaces that were presented in Section 2.

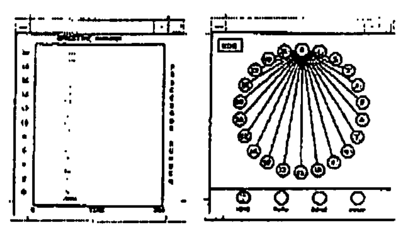




Event domain views using Matlab as an analytic and visualization engine, integrated with VIZIR.



VIZIR provides control and synchronization of the environment.



Typical application visualization displays from ParaGraph integrated with VIZIR

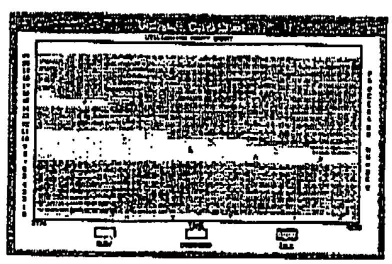


FIGURE 3. Multiple views from multiple domains and tools to visualize and analyze on-the-fly execution behavior of a distributed program.

3.3 Multiple Views using Multiple Tools

Visualizing the behavior and performance of a parallel or distributed program is always a multi-dimensional assessment. Not only does it take multiple tools but also multiple views and multiple-domains [11] are needed to represent a complete picture of program behavior to the user. VIZIR provides typical performance visualization and animation views that are implemented in ParaGraph. It can also allow the use of general-purpose data analysis tools such as AVS, Matlab, Gnuplot, Mathematica, and so on to represent multiple perspectives on application performance and behavior. Figure 3 represents some of these views.

4.0 Conclusions

VIZIR is an on-going experiment in Hewlett-Packard Research Labs. VIZIR provides standard window interfaces to existing visualizers/debuggers with on-the-fly control and synchronization. Processes in the parallel application can be halted by the debugger at the same point that performance and data visualization is being done. In addition, For example, performance and data errors detected by performance and data analyzers may automatically trigger the debugger to halt the process.

Although we have applied our mechanisms to prototype debugging environment for parallel programs, they have much wider applicability. This approach can be used anytime we want to do the same thing on more than one machine. Examples include administering cluster, sharing large volume of data such as video.

Acknowledgment & References

Thanks to Chris Hsiung and Diane River from the Michigan State University for their encouragement and suggestions. Vineet Singh and Milon Mackey also provided useful technical discussion.

[1] Beguelin, A. L., "Xab: A Tool for Monitoring PVM Programs," Proceedings of the Twenty-Sixth Hawaii Int. Conf. on System Sciences, Wailea, Hawaii, Jan. 1993, pp. 102–3.

[2] Chen, Minder and Ronald J. Norman, "A Framework for Integrated CASE," IEEE Software, March 1992, pp. 18–22.

[3] Geist, G. et al., "A Machine-Independent Communication Library," Proc. of the Fourth Conf. on Hypercubes, Concurrent Computers, and Applications, Los Altos, 1990.

[4] Geist, G. et al., "PVM 3.0 User's Guide and Reference Manual," ORNL/TM-12187, February 1993.

[5] Hao, Ming C. Alan Karp, Milon Mackey, Vineet Singh, Jane Chien, "On-the-Fly Visualization and Debugging of Parallel Programs", Proceedings of MASCOTS '94, Durham, North Carolina, Jan. 31-Feb. 2, 1994.

[6] Hao, Ming C. Alan Karp, Vineet Singh, "Concurrent Application Control in Collaborative Computing," HPL-94-37,Hewlett-Packard Laboratories, Palo Alto, 4/94

[7] Hao, Ming C. Alan Karp, Mehdi Jazayeri, "MESH: Sharing Multiple Events in Distributed Computing" HPL, Hewlett-Pack Laboratories, Palo Alto 7/94.

[8] Heath, Michael T. and Jennifer A. Etheridge, "Visualizing the Performance of Parallel Programs," IEEE Software, 8(5), September 1991, pp. 29–39.

[9] Kraemer, Eileen and John T. Stasko, "The Visualization of Parallel Systems: An Overview," Journal of Parallel and Distributed Computing, 18(2), June 1993, pp. 105–117.

[10] Pancake, Cherri M., "Supercomputing '93 Parallel User Survey: Response Summary," 94–80–2, Oregon State University, Feb. 1994.

[11] Rover, Diane T. and Abdul Waheed, "Multiple Domain Analysis Methods," Proc. of the 3rd ACM/ONR Workshop on Parallel and Distributed Debugging, May 1993, pp. 53–63.

[12] Waheed, A., B. Kronmuller, Roomi Sinha, and D. T. Rover, "A Toolkit for Advanced Performance Analysis," proceedings of International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '94) Tools Fair, Durham, North Carolina, Jan. 31– Feb. 2, 1994.

Concurrent Application Control In Collaborative Computing

Ming C. Hao, Alan H. Karp, Vineet Singh

Jane Chien

HPL–94-37

Event sensing,

Concurrent control,

Multicasting window events

Distributed collaborative

Heterogenous

Most people think that collaboration implies that several people are sharing work on a single application with shared displays. In fact, collaboration is more. It includes the concurrent control of multiple applications by a collaborative group. To enable this more powerful form of collaboration, we show how to combine earlier mechanisms for single client, multiple server computing with a new mechanism called ESP (Event Sense Protocol) for multiple client, single server computing. We describe two extended examples — a working prototype of a multi-user, heterogeneous, parallel debugger and a commercial banking application.

1.0 Introduction

Today's collaborative computing environments do not address collaboration using multiple applications executing concurrently. There are many example of systems that allow collaboration using a single application on a single data repository. For examples, HP's SharedX [3] product allows sharing of an X protocol based application among users in a distributed computing environment. MMConf [4] provides the Diamond Multimedia conferencing System. BBN's Slate [5] allows users to collaborate on document development. However, all the above conferencing systems are limited to sharing a central copy of an application. There is no mechanism to concurrently control more than one application or more than one data repository. The missing piece is the subject of this presentation.

We have developed a single server, multiple client environment. ESP, Event Sense Protocol[1], allows a user to control several clients (applications) simultaneously. A key feature of our system is that any existing application can be used with no modification of any kind. For example, ESP enables us to update multiple copies of a Lotus spreadsheet by entering the commands once. In fact, the applications being controlled need not be running on machines of the same architecture or even be identical applications. ESP allows us to control Lotus running on HP and Sun workstations and Excel on an IBM system by typing commands once. The only requirement is that the commands typed be meaningful to each system.

This kind of multiple application, multiple data repository collaboration has many uses. A corporate financial officer could update divisions' independently held spreadsheets to reflect changes in Federal tax law and leave the local revenue numbers unchanged; division heads may change local numbers concurrently. A secretary could change the boiler plate part of standard letters used by all the Company's branch offices.

Combining the conventional single client, multiple server collaborative environment with the single server, multiple client model of ESP completes the picture by providing for multiple application collaboration. Application experts in different locations could simultaneously control various aspects of a distributed application running on several machines. An instructor teaching FrameMaker could control many student documents simultaneously to illustrate a particular point while still allowing the students to work on their own reports and commenting on their content. Systems support people could work with users to improve distributed applications.

The next section describes ESP, a mechanism to enable concurrent application control. Section 3 shows how ESP can be integrated into a collaborative computing environment. Section 4 adds a couple of extended examples of multi-user, multi-application collaboration. Section 5 discusses related work and Section 6 presents our summary.

1. Patent pending

2.0 ESP — A Mechanism to Enable Concurrent Application Control

The current graphical user interfaces GUIs [7] are based on a single-threaded dialog, where the user operates on one single command button to invoke one application and to execute one single function at a time. There has been a need to have a mechanism to sense user commands and dialogs, then to control, manage, and multicast them to a set of applications for executing some functions simultaneously. ESP provides this mechanism without requiring any modification to existing system or application software in a distributed environment.

ESP has the following features:

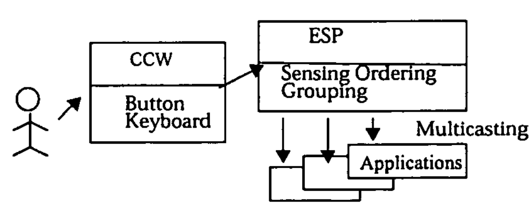
- Built on a multiple client-single server model and uses standard window interface. There are no special libraries, and there is no need to compile or even re-link the existing application source code. This mechanism should be portable to all window systems.
- Provides event sensing and multicast mechanisms to make selected processes execute concurrently.
- Provides a concurrent dialog to a group of applications to execute multiple functions simultaneously.
- Provides layered GUI interface to allow graduated access to the more powerful and complicated GUI capabilities.
- Provides a simple point-and-click approach to dynamically connect or disconnect selected existing applications in a working context.
- Provides dynamic context for multicasting.
- Provides dynamic button assignment for heterogeneous applications.

2.1 ESP Architecture

ESP is built on a multiple client-server model. Figure 1 illustrates the overall architecture. It contains two key components:

1. Concurrency Control Window (CCW)
2. Event Sense and Muticasting Processing (ESP)

FIGURE 1.ESP Architecture



Applications use windows to communicate with users. Users request an application to perform a function by sending events, such as a result of a key, mouse button, or sprite motion. The application GUI behavior understands its window events and how to interact with the application. This mechanism is started by sensing all the possible events happening on an application window.

In general, there are two types of windows commonly used by applications to interact with users: buttons and text fields. The event type received on these windows are button press, button release, key press and mouse movement. Our mechanism senses the common events and puts them into the following two categories: keyboard events and mouse events.

Both keyboard and mouse events contain key press/release and button press/release. Both events are sent by the server to the application when the user presses a key on the keyboard or presses the mouse. Only an application that has specifically asked to be informed of that type of event will receive them.

ESP senses and selects the above standard common window events as the potential candidates to be managed and multicast. However, additional events may need to be included for processing special types of new application windows.

2.2 ESP Operation

ESP operation comprises the following four major steps:

2.2.1 Access Running Application Windows

The purpose of accessing running application windows is to find the window characteristics, their child windows, and their identifiers. This mechanism uses window query tree system calls, window images, or pointer movement to find the corresponding application window structure and its identifiers. Normally, buttons and text fields are the children of the application main window. Buttons and text fields are the input windows used by users to interact with the application.

2.2.2 Build CCW and Child Windows

This mechanism builds the Concurrency Control Window (CCW) to control and manage the activities in the event sensing and multicasting session. In order to receive the same event types, the CCW has to simulate/build the same type of child windows as the application, such as a button to represent the application's button, a command line input field to represent the application text window.

2.2.3 Sense User Window Events

Using a CCW (Concurrency Control Window), this mechanism senses the user actions/window events and passes them to the corresponding application windows to execute some function concurrently. We use Xt Intrinsics multiple input application contexts and registered event handlers in our current X Windows prototype. Similar methods can be applied to other window systems.

2.2.4 Multicast User Events

After building the same type of window, ESP controls and multicasts the window events received from the end user to the selected windows to execute a user command such as simultaneously changing the sales tax rate in all participating spreadsheets. Using this mechanism, the user can concurrently manage the existing application's execution.

There are two modes of operation:

1. Global:

   Using CCW window global buttons to invoke all or a selected group of the running applications to perform a function simultaneously.

2. Local:

   Using CCW's individual selection buttons to raise a specific window for executing an independent function.

Methods for using these modes include the following:

In the global mode, the application GUIs may be structured in a hierarchical fashion with an arbitrary number of levels. Interacting with a GUI leads to window events being generated on one or more of the children GUIs. The mapping of window events to children GUIs is many-to-many. The children applications may be sequential or concurrent. Window events on children GUIs lead similarly to further events being generated on grandchildren GUIs. As an example of multiple events being generated for a single child GUI, a show button on the CCW window could be mapped to both the data visualization and performance visualization buttons on the same child visualization application.

Child GUIs may be specialized to different functions available in different applications or even different components of the same sequential application. In the local mode, ESP allows the independent construction of the different functional GUIs. A user can interact with the CCW window and access some generic functionality or access the individual children GUIs for access to specialized functions. For example, a parallel application could be constructed out of a Lotus 123 spreadsheet application, Pablo [2] (a performance visualization application), and a mail program in order to do some spreadsheet calculations on raw performance data, graph computed visualization data, and mail it to a distribution list. Specialized GUIs are provided by the individual application GUIs whereas generic and composite functionality is provided by the CCW window.

2.3 ESP Operations Extensions

ESP operation includes the following three extensions:

2.3.1 Dynamic Window Association

ESP can associate the generic windows, such as buttons, text fields, with a specific window in the same or different applications. In addition, a group of application buttons can be associated with a generic CCW button. With ESP, users can dynamically associate their CCW buttons with any heterogeneous function they want to perform at run time.

2.3.2 Event Concurrency Control, Grouping, and Synchronization

ESP uses a layered structure to concurrently control the selected events happening in the existing application GUIs. One single user request may generate multiple dialogs with the applications which depend on the structure of the application GUIs.

The user can dynamically add or delete an application window to or from the working context. Each application window is only used for processing local application functions.

ESP allows the user to synchronize different types of window events to manage the order of execution by the applications. When associating the application buttons with a CCW button, the user can indicate the order of execution and which group of functions can execute simultaneously.

2.3.3 Enabling Full-Screen Applications

This mechanism can be also used on the applications that allow the user to interact with the screen anywhere on it. Example interactions include text insertion, button press etc. We make our CCW an exact replica of the application window. Putting the replica directly under one of the instances of the application window and setting the focus to the replica, makes it looks to the user like everything typed in one application window is simultaneously being typed in all instances of the application in the current context.

3.0 Integration of ESP with Collaborative Computing

Human endeavors involve more than one person, especially in a complex distributed and parallel application. Different expertise is need to work on different parts of a program. At times, it is appropriate to examine a single running instance of an application with multiple experts. For example, we can use HP SharedX to allow more than one person to observe and control the program on a remote workstation. However, there is no global control mechanism to allow users to control concurrently the interactions with a set of shared applications.

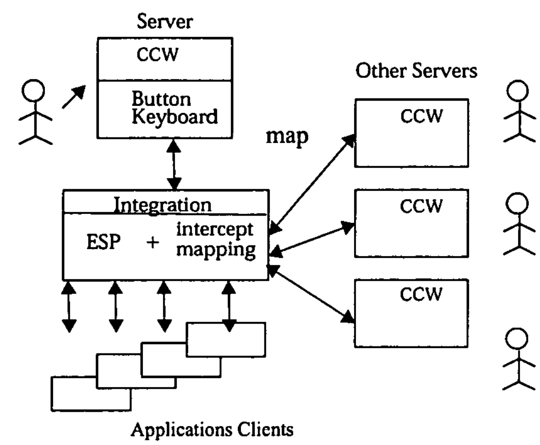
ESP solves the problem of how to sense user actions, to control, and to distribute input window events to each selected application window for simultaneous execution in a distributed environment. The integration of ESP with collaborative computing achieves the goal of sharing multiple applications among users. It contains three key components:

1. Concurrency Control Window (CCW)
2. Event Sense and Muticasting Processing (ESP)

3. Intercept requests from application clients and map them onto the other servers [11]

Figure 2 illustrates the integration of ESP with collaboration.

FIGURE 2.An Integration of ESP and Collaboration



4.0 Two Examples

The examples presented here show how potential applications might use ESP. The first example is IVD [1], a Hewlett-Packard Laboratories Interactive Visualization Debugger working prototype for message passing parallel applications.

The second example is a commercial banking application that coordinates multiple spreadsheets, a plotting program, and a mail program.

4.1 A Parallel Visualization Debugger

PVM (Parallel Virtual Machine) [6] is a popular system for writing parallel applications. PVM requires a debugging interface that will support debugging functions for all selected processes using a single window. Such an interface will be very valuable for actions such as simultaneous single-step execution in all selected instances. At present, PVM uses a separate window for each selected process. Users have to enter commands in each debugging window.

IVD uses ESP to integrate existing debuggers and visualizers to debug/observe parallel application execution behavior in a distributed, shared-nothing multicomputer environment with the following features:

4.1.1 Heterogeneous processing

ESP is designed for heterogenous processing. Using ESP input event sense and multicasting capability enables IVD to access existing debuggers across various platforms to debug large, complex problems.

4.1.2 Scalability by grouping

ESP is scalable. Each process has its own debugging and visualization facilities. ESP provides different "contexts" for the user to dynamically select the multicasting scope. By sending commands only to processes within the single, currently active context rather than all processes simultaneously, the user may limit the amount of overhead generated by running multiple processes.

With a context, the user can group certain debugger instances together. Each context then receives the same debugger commands entered on the IVD concurrency control window. By grouping the debugger instances, the user can indicate the execution order of groups of debugger instances. The user is not restricted to work only with individual debugger instances. This becomes useful when certain debugger instances are naturally associated with one another. For example, one context may contain all the slaves while another context contains only the master. Alternatively, there may be many contexts, each containing the debugger instances that are functionally related to one another. A given debugger may belong to more than one context or to no context.

4.1.3 Dynamic Context Modification

Figure 3 illustrates the mechanism to dynamically group a set of application programs to run simultaneously.

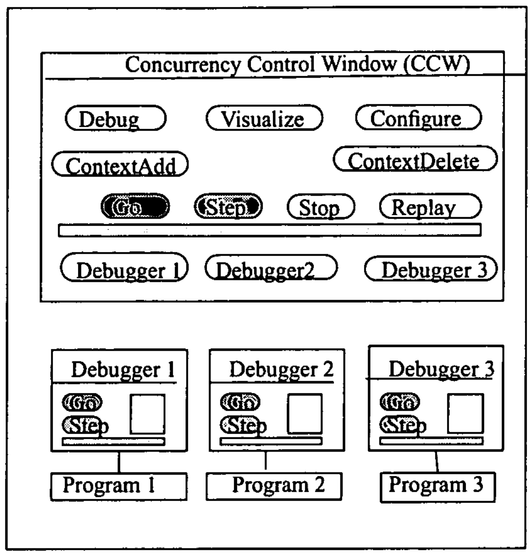
A user starts a visualization and debugging session in a parallel programming environment. The user starts up the session by bring up a Concurrency Control Window. Afterwards, the user performs the following actions:

1. Presses the "Debug" button in CCW to start the debugging environment and bring up the "Debugger 1" window.
2. Types "debug program 1" in the debugger window command line.
3. "Program 1" spawns "program 2" and "program 3" running under "debugger 2" and "debugger 3" respectively on different workstations.
4. ContextAdd debugger1, debugger2, and debugger 3 in the current debugging context.
5. Uses global buttons on the CCW window to issue go, step and other debugger command.
6. Types setting breakpoint, print commands in the multicast command line.
7. Steps through application 1, 2, 3 concurrently to locate the problem.
8. Press global "step" button to simultaneously single-step through the programs 1, 2, 3.
9. ContextDelete "Debugger 2" from the working set. Step through programs 1 and 3.
10. ContextAdd "Debugger 2" to the current working set.

11. Types "print a" to ask programs 1, 2, 3 to print the variable "a" in their windows simultaneously.
12. Press "Go" button to ask program 1, 2, 3 to continue execution simultaneously

Repeat the above global/local debugging operations until the bug is found.

FIGURE 3.Dynamic Context Modification

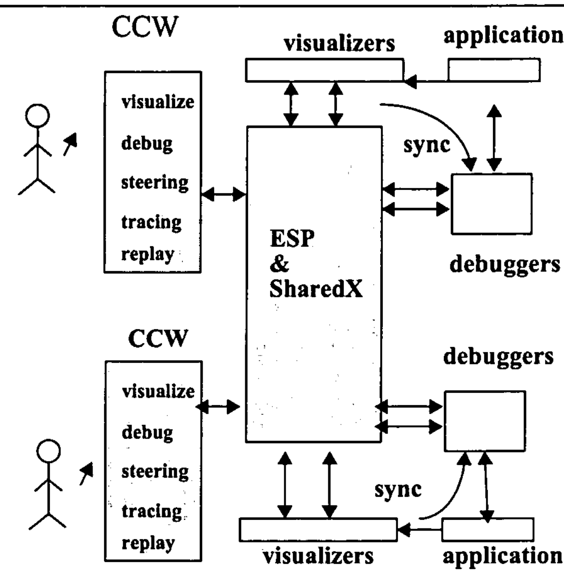


4.1.4 A Multi-User collaborative Debugging Session

Figure 4 shows that IVD with ESP appears as a single visualization debugger to the user. ESP manipulates the user input events, such as keyboard and mouse, and then multicasts these events to each running visualizer and debugger. ESP automatically triggers the visualizers/debuggers to execute received events from the user. User events are processed as if the window events had been directly entered into the visualizer/debugger windows.

HP SharedX allows more that one person to share and control a single running instance of multiple applications including IVD. Hence a systems expert and the program developer can work together to fix the program.

FIGURE 4. Collaborative Debugging

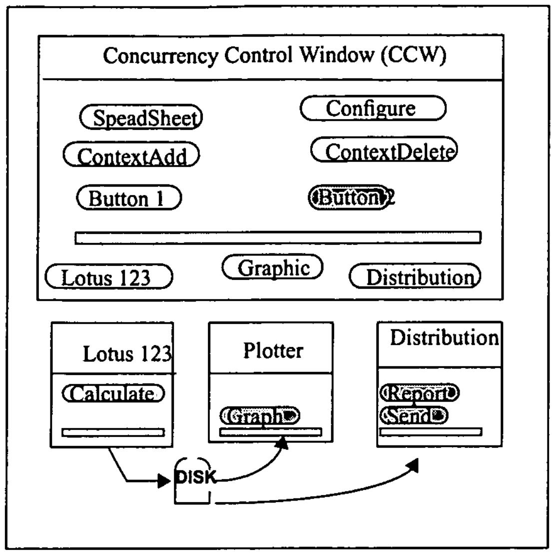


4.2 A Commercial Banking Application

Consider a group of departments that run a spreadsheet each month, plot some graphs, and print a report. The normal mode of operation is to have someone build the spreadsheet templates including formulas, set up the graphics formats, and build the reports. These are distributed to machines in each department. Secretaries in each department enter the department specific data into the work space and run the graphs and reports. Any changes to the spreadsheet formulas or report formats must be distributed to each department and updates made locally in a timely manner.

With our mechanism the procedures would be somewhat simpler. The person responsible for maintaining the spreadsheet would bring up a control window. The spreadsheet application would be started on each node using the facilities of the control window, and a flag would be set to indicate that changes made to one machine's copy would be made to all. Next, typing in one window, the user would build the spreadsheet, graphics formats, and reports. Changes would be made the same way. The department secretaries still enter the department specific data. At the end of each month, one person could bring up the spreadsheets on all the machines, set the context so that commands are globally executed, and generate all the graphs and reports by typing the commands in only one window.

FIGURE 5.A Banking Application Association



An extension to this example is to use ESP to control completely different applications. The bank uses spreadsheets to calculate its annual result, then uses the specific graphic and distribution programs to plot, produce reports, and send to all the customer. The generic "Button 1" on the CCW associates with Lotus 123 "calculate" button. The generic "Button 2" associates with three buttons: "Graph" on the graphic program window, "Report" and "Send" buttons on the distribution program windows.

It only takes the accountant two steps to distribute the annual reports:

- Press the "Button 1" to compute the annual income and expense. The results are stored on a disk.
- Press the "Button 2" button to read data from the disk, make the graphic charts, produce the annual reports, and distribute the reports to the customers. All three steps are performed by pressing the "distribute" button and all three functions are executed concurrently. ESP allows the user to define the ordering of the event to be sent.

5.0 Related Work

Research in the area of asynchronous collaborative computing has been published under a variety of subjects in the recent past.

The Lotus Notes [8] product allows users to get instant access to the same information without any further intervention by sender or the receiver. This is similar to our commercial banking example using multiple, shared spreadsheets. There are two differences, however. First, ESP allows multiple users to manipulate multiple applications directly and not just update data files.

Second, Lotus Notes uses a relational database whereas ESP multicasts uninterpreted window events directly to application windows.

A couple of other systems actually allow concurrent manipulation of multiple applications by multiple users. A system from IBM[9] uses global cursor for concurrent data entry and manipulation in multiple applications. A system from Wang Lab [10] routes keystrokes to processes which are associated by reference to a a routing table. However, most of these concurrent control of multiple applications have limitations. IBM and Wang's mechanisms require cursor position tracking and keystroke interpretation while inputting common data into a plurality of programs. The overhead cost is very high in both. ESP provides a concurrency control window to receive, order, group and manage incoming window events and to forward them to running applications. ESP does not interpret keystrokes and pointer movement. Further, this mechanism can access existing application GUIs at run time without changing source code. There is no recompilation, or re-linking, and no special library is needed

ESP can be used in many situations. We have already shown that it is the foundation on which to build an interactive, parallel, visualization debugger in a distributed and parallel environment. We have also noted that this event sense, control, and multicast ESP allows us to mix different application GUIs, such as to combine four applications' buttons — calculate, graph, report, and send — to generate a report and mail it. In addition to being used in a parallel debugging session, this ESP is directly applicable in the following applications:

1. Multiple database queries and updates
2. Networking and manufacturing control
3. Multiple shadow file creation and updates
4. Multiple updates for spreadsheet and banking applications

In general, ESP applies to all GUI applications. It allows us to take any application GUI and to make it distributed. It allows the end user to control and manage multiple dialogs with existing applications without impact on the existing system or application software. In addition, with SharedX, ESP makes collaborative computing possible to control existing applications concurrently in a distributed heterogeneous environment. ESP provides a singe image in a multi-user, multi-application distributed environment.

6.0 Summary

ESP provides unique features not found in other collaborative computing systems as shown in the previous sections. IVD uses ESP to provide an on-line interactive visualization debugging tool. In addition, ESP has the ability to group processes into various contexts and perform simultaneous operations. Programmers can use their favorite tools. The most interesting feature is that many different, unmodified tools can be accessed simultaneously by multiple users in a distributed

shared-nothing environment. With ESP enabling technology as a vehicle, we will continue to explore collaborative computing with multiple applications.

7.0 Acknowledgment

Thanks to Dr. Chris Hsiung and Dr. Mehdi Jazayeri for their encouragement and suggestions, and to Charles Young and Don Carfinkel for their technical discussions.

8.0 References

[1] Ming C. Hao, Alan H. Karp, Milon Mackey, Vineet Singh, Jane Chien "On-the-Fly Visualization and Debugging of Parallel Programs", Proceeding of the IEEE/ACM MASCOTS 94, January 31, 1994, Durham, North Carolina.

[2] Pablo Instrumentation Environment, Department of Computer Science, University of Illinois, Urbana, Illinois, 1992

[3] John R. Portherfield "Mixed Blessings" and HP "SharedX" HP Professional Volume 5, Issue 9, HP SharedX Unix Today, May 27, 1991.

[4] Terrence Crowley and Harry Forsdick, "MMConf: The Diamond Multimedia Conferencing System", BBN System and Technologies, INC. Aug 23, 1989

[5] BBN/Slate, BBN Software Products, Cambridge, M.A. 1990

[6] G. A. Geist, "Network Based Concurrent Computing on the PVM System." Mathematical Science Section, Oak Ridge National Laboratory, Oak Ridge, TN 3781. 1992.

[7] Communications of the ACM Special Section on "Graphical User Interfaces: The Next Generation", April 1993, Volume 36, Number 4

[8] Lotus Notes Users' Guide, Lotus Development Corporation, Cambridge, MA, 1989.

[9] Robert J. Torres, "Method for concurrent data entry and manipulation in multiple application", IBM patent application, September, 1989.

[10] Deborah A. Rhodes, Waltham Mass, Eric Rustici, Kelly H. Carter, "Method for routing events from key strokes in a multi-processing computer system", January, 1992, Wang Laboratories, Inc.,

[11] M. C. Hao, F. D. Snow, J. Chien, J. A. Latone, "Experiments in Collaboration with Existing X Applications", G32-3569, May, 1992, IBM Palo Alto Scientific Center.

OIPE IAP69 OCT 31 2006 PATENT & TRADEMARK OFFICE

Notice of Allowability	Application No. 09/904,300	Applicant(s) SHAFRON ET AL.
	Examiner LaShonda T. Jacobs	Art Unit 2157

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *October 7, 2005*.
2. ☒ The allowed claim(s) is/are *1-31,33-61 and 63-84*.
3. ☐ The drawings filed on _____ are accepted by the Examiner.
4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
   1. ☐ Certified copies of the priority documents have been received.
   2. ☐ Certified copies of the priority documents have been received in Application No. _____.
   3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.
6. ☒ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
   1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☒ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date 6.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).
7. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☒ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☒ Information Disclosure Statements (PTO-1449 or PTO/SB/08), Paper No./Mail Date October 7, 2005
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application (PTO-152)
6. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____.
7. ☐ Examiner's Amendment/Comment
8. ☒ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

ARIO ETIENNE
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

DETAILED ACTION

Response to Amendment

This Office Action is response to Applicants' RCE filed on October 7, 2005.

Information Disclosure Statement

1. The information disclosure statement (IDS) submitted on October 7, 2005 was filed after the mailing date of the Notice of Allowance on August 26, 2005. The submission is in compliance with the provisions of 37 CFR 1.97. Accordingly, the information disclosure statement is being considered by the examiner.

Reasons for Allowance

1. The following is an examiner's statement of reasons for allowance: the closest prior art of record (Shelton et al., U.S. Patent No. 5,954,798) and (Gavrilescu et al., U.S. Pub. No. 2002/0198941) does not teach or suggest in detail controlling Internet navigation of the second computer based upon Internet navigation of the first computer, wherein the first script and the first control and the second script and the second control are independent from Web pages that are displayed on the first on the first computer and the second computer in combination with all the elements of the independent claims as argued by Applicants. So as indicated by the above statements, Applicants' arguments have been consider persuasive, in light of the claims limitations.

2. The dependent claims further limit the independent claims and are considered allowable on the same basis as the independent claims as well as for the further limitations set forth.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

3. Claims **1-31**, **33-61** and **63-84** are allowed.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to LaShonda T. Jacobs whose telephone number is 571-272-4004. The examiner can normally be reached on 8:30 A.M.-5:00 P.M..

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ario Etienne can be reached on 571-272-4001. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

ltj
October 18, 2005

LaShonda T Jacobs
Examiner
Art Unit 2157

ARIO ETIENNE
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100